

I've been told that DASADA is a town in the home state of Mahatma Gandhi. This seems a fitting name for the program, since today's military missions that include both peacekeeping and war fighting.

Despite the euphonic name, the words in the program title actually do describe what we're trying to do:

Dynamic Assembly means that we can change system components, connections, or topology at run-time.

Adaptability includes both "semantic interoperability" -- making sure we're using the same names for the same things -- and predictability -- making sure components work correctly when we put them together.

Dependability and Assurance are things the commercial market pays less attention to than what DoD needs for its systems.

To get the systems we need, we need gauges to measure what the system is doing.

Note that the industrial revolution was enabled by improvements in measurement -- we had punch presses and lathes beforehand, but it was measurement that enabled interchangeable parts and assembly lines. We need similar ways of measuring software products for composability.

We need to be able to use these measures to guide system evolution by updating our understanding of what the system is doing in comparison with what we think it should or should not be doing.

As systems get more complex, they become harder to understand. System integration problems with the Navy's Cooperative Engagement Capability software are going to take several hundred million dollars to fix.

It should be obvious that if we can't measure what's going on in a system, We can't model it, We can't understand it; We can't Predict it; We can't Control it; And We can't Automatically adapt it to meet new situations.

From a technical point of view, typical reasons why components don't work together include:

- * Interfaces don't pass the right information, (and)
- * Modules make assumptions, but don't tell the rest of the system, (and)
- * Timing constraints are not stated.

DASADA is critical to future DoD and commercial systems. Systems (of systems) are getting more difficult to understand, build, operate, and evolve. We have fewer trained people who can understand, build, operate, and evolve them.

Currently, industry has little incentive to fix these problems. Major vendors support interoperability, as long as it's with their own products. They build their market through product differentiation, not integration.

DASADA's architecture-based approach to predictable, dynamic, component composition should provide solutions. It will help us gauge important software properties, so we can get software components to work together predictably.

The architecture-based approach will help us reduce cycle time by helping us:

- * Dynamically assemble, reconfigure, and evolve systems.
- * Easily introduce new components to add functionality.
- * Adaptively and dynamically scale systems, and,
- * Continuously upgrade components

We've got a very short video showing how an architecture representation can ensure that design constraints are upheld - in this case, guaranteeing that two processors process the same message.

(Aegis Video)

Architectures model component interaction to guide system transformations. These transformations can include adding, deleting, or replacing either a component or connection.

For example, suppose we need to add secure communications to a system. The system could access dependency models to determine what type of modifications are needed and how to carry them out.

The architectural model could help:

- * Identify modules that need to be changed to incorporate cryptography software.
- * Dynamically model the interaction of cryptography components with the timing of the underlying applications to ensure performance and freedom from deadlock. And,
- * Compose the needed communications infrastructure.

The questions DASADA is trying to answer are "Which transformations are correct with respect to system requirements and constraints?"

"Which transformations are "best" with respect to ensuring critical properties?"

Architecture notations model configurations, components, connectors, events, and constraints.

Configuration gauges measure component interactions with respect to properties such as quality of service and liveness. Do components communicate? And, How often?

Component gauges assess whether components are compatible with respect to the functions they perform and the data they consume and produce.

They assess whether all or part of the component's functionality is being used, helping to identify dead code.

Connector Gauges evaluate the dynamic behavior of connections and determine if a replacement connector is compatible with the existing infrastructure.

Event Gauges evaluate component interaction protocols and usage patterns. Deadlock situations occur because of component misunderstandings about who is supposed to initiate or terminate operations - event gauges should detect this.

DASADA is developing gauges to measure important software properties to ensure that software components work together. It's looking at how to gauge interoperability throughout the evolution cycle, addressing challenges involved in 3 stages:

Continual Design gauges assess component and connector suitability before assembly, allowing automated assembly and on-the-fly transformations that produce predictable, safe systems.

Continual Coordination gauges assess component suitability during assembly, allowing reconfigurations to be conducted safely across heterogeneous, distributed dynamic systems. Continual coordination emphasizes the sequence in which changes are made - remembering, for example, to back up persistent data before deleting a node.

Continual Validation gauges assess suitability after assembly, providing continual, run-time validation of critical system properties.

The following slides demonstrate a few of these gauges and the infrastructure that's being developed to support their use.

These gauges verify that system architecture meets design and component resource requirements.

In this example, we refine a system specification by selecting an operating system. (text segment switch)

Linux in this example.

This choice may place constraints on the behavior of the system in terms of power and cost.

This may, in turn, affect our freedom of choice with respect to processors or routers.

The technical basis of these gauges are constraints specified in the system architecture.

It's important to measure the actual run-time configuration and interaction of components in dynamic systems, since these can't always be predicted in advance.

We need to measure the time-varying connectivity of components so we can see what components are actually being used and so we can improve linkages where needed.

We also need to measure other aspects of component interaction -- when components communicate, which operations are invoked, how much data is exchanged, how long responses take, and what exceptions occur and under what circumstances.

As you can see at the lower right, this can help us find dead code.

The "Evolution and Integration Command Center", will integrate and analyze gauge readings to ensure that components behave as expected during dynamic system evolution, integration, and re-configuration.

It is based on an XML-based event description.

It uses architectural models to automate the insertion of probes and the generation of gauges to guarantee specified properties. It enables "go/no-go" decisions about re-configuration alternatives and monitors the "live" evolution of the system.

Previous slides have provided examples of the technical developments we're expecting from DASADA. We've completed selection for technology development efforts, which are now underway.

A planned second phase of the program involves larger experiments, to be conducted in collaboration with DoD Systems offices.

It is anticipated that funding will be available in FY2001 and FY2002 for these organizations, and their contractors, to begin planning for experimental demonstrations in FY02 and FY03. These planning funds are intended to partially support efforts to:

- * First, identify systems/subsystems on which experiments will be conducted.
- * Second, evaluate technologies and combinations of technologies to be used in the experiments.
- * Third, define evaluation criteria and measures and available baseline data. And,
- * Fourth, plan experiments.

We plan to conduct two to three experiments, which will be funded at a level sufficient to provide meaningful results to potential DoD customers for transition planning.

We thought we'd show you some of the technology developed in a precursor program - the Evolutionary Design of Complex Systems or EDCS.

First is the use of architecture models to reduce integration time/cost. This is illustrated by a dramatic reduction in the time required to set up tests at the Arnold Engineering Development Center.

(ITIS Video)

Second is using semi-formal architecture description languages to guarantee critical system properties - in this case some tools that are tailored for control systems design and analysis.

(Honeywell Video)

Third is an animation explaining where we're going in testing and assurance. We could show an actual video, but demonstrations that show nothing going wrong, because you've fixed all the problems, tend to be REALLY dull. This segment portrays three testing scenarios:

- * First, The recent past, using a single holey net at the end of development.
- * Second, Current advanced practice, using multiple techniques throughout the life cycle. And,
- * Third, Current research approaches that combine testing and analysis with fault tolerance.

(Bugs Video)

DASADA is using architectural notations and gauges:

- * To -- Support run-time dynamism (modification.)
- * To -- Model dynamic systems.
- * To -- Monitor constraint satisfaction, both during design refinement and during operation.
- * To -- Integrate multiple views of evolving designs with system management functions. This will provide adaptive management and self-correction.
- * And, finally, to demonstrate architecture- and model-based tools that assure component interoperability in dynamic systems.

We invite your involvement in this program. Please visit our web site.

We look forward to talking with you.

Thank you for your attention!